Indonesian Journal of Electrical Engineering and Computer Science
Vol. 12, No. 1, October 2018, pp. 78~86
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v12.i1.pp78-86

Fast Ant Colony Optimization for Clustering

Abba Suganda Girsang[1], Tjeng Wawan Cenggoro*, and Ko-Wei Huang[2]
[1]Computer Science Department, BINUS Graduate Program - Master of Computer Science,
Bina Nusantara University, Jakarta, Indonesia
[2]Department of Electrical Engineering, National Kaohsiung University of Science and
Technology, Kaohsiung City, Taiwan, R.O.C

Article Info

Article history:

Received Jan 10, 2018
Revised Apr 21, 2018
Accepted Jun 14, 2018

Keywords:

Pattern Reduction
Ant Colony Optimization
Fast Ant Colony Optimization
Clustering
Meta-Heuristic Algorithm

ABSTRACT

Data clustering is popular data analysis approaches, which used to organizing data into sensible clusters based on similarity measure, where data within a cluster are similar to each other but dissimilar to that of another cluster. In the recently, the cluster problem has been proven as NP-hard problem, thus, it can be solved with meta-heuristic algorithms, such as the particle swarm optimization (PSO), genetic algorithm (GA), and ant colony optimization (ACO), respectively. This paper proposes an algorithm called Fast Ant Colony Optimization for Clustering (FACOC) to reduce the computation time of Ant Colony Optimization (ACO) in clustering problem. FACOC is developed by the motivation that a redundant computation is occurred in ACO for clustering. This redundant computation can be cut in order to reduce the computation time of ACO for clustering. The proposed FACOC algorithm was verified on 5 well-known benchmarks. Experimental result shows that by cutting this redundant computation, the computation time can be reduced about 28% while only suffering a small quality degradation.

Corresponding Author:

Ko-Wei Huang
Department of Electrical Engineering,
National Kaohsiung University of Science and Technology
No. 415, Jiangong Rd., Kaohsiung City, Taiwan, R.O.C
Email: elone.huang@nkust.edu.tw

1. INTRODUCTION

Clustering is a problem which goal is to find hidden structure behind dataset. It is done by grouping the data into several cluster by the means of their similarity. Clustering falls into the category of optimization problem as it needs to estimate the optimal position of an unknown cluster center. Unlike other optimization problem, clustering plays a wide role in the recent development of computer science [1]. It has been found capable to be applied in various field: data mining [2-5], image processing [6-8], geographical information system [9-12], computational biology [13, 14], road sccidents analysis[15], routing protocol [16], and power consumption [17], respectively. Because of its significance, a lot of researches has been conducted in order to improve the performance of clustering algorithms.

One of the performance indicator that needs to be improved in clustering algorithm is its computation time. As we enter the era of big data, the amount of retrieved data is growing massively. Analyzing those data with traditional clustering algorithm may not feasible in considerable amount of time. Thus, it is essential to improve a clustering algorithm so that it is time-efficient. Because of its importance, several researches has been conducted to improve computation time in various clustering algorithm [18-25].

The focus of this paper is on improving the computation time of Ant Colony Optimization (ACO) for clus-tering. Compared to other traditional clustering algorithm such as Simulated Annealing, Genetic Algorithm and Tabu search, ACO has been found to have a better result in quality [26]; thus the improvement will create a clustering algorithm with high results quality and reasonable computation time.

The strategy for reducing the computation time employed is pattern reduction. Pattern reduction seeks for redundant process in the algorithm and remove it to de-crease computation time. Pattern reduction is adopted because in ACO for clustering, redundancies in generating new solutions are observed. In the rest of this paper, the proposed algorithm will be called as Fast Ant Colony Optimization for Clustering (FACOC) for simplicity. The remainder of this paper is organized as follows. Section 2 provides the background information. The proposed FACOC algorithm is presented and evaluated in Sections 3 and 4, respectively. Brief conclusions are presented in Section 5.

2. RESEARCH METHOD

2.1. Ant Colony Optimization

Ant colony optimization (ACO) is a meta-heuristic optimization algorithm that inspired by the behavior of real ants [27]. ACO originally developed as an algorithm to solve travelling salesman problem, introduced as Ant System (AS) [28]. In 1997, the ant system improved to control its exploration and exploitation, introduced as Ant Colony System [29]. The basic ACO algorithm is shown in algorithm 1.

Algorithm 1 Pseudocode for ACO algorithm
initialize
While stopping condition is not met
generate new solutions
update pheromone
local search
EndWhile

ACO initializes all of the initial parameter such as initial pheromone (t0) and evaporation rate (r). The main body of the algorithm is repeated until stopping condition is met, shown in the 3rd to 5th line in algorithm 1. Then, the algorithm repeated its main body until stopping condition is met.

The first step of the main body is to generate new solutions. In generating new solution, ACO calculates the probability of each sub-solution based on the total pheromone that has been laid in it. For TSP, the probability is calculated as shown in Equation (1)

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{j \in allowed_k}[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}} \quad (1)$$

In Equation (1) τ_{ij} is the pheromone value for edge between city i and j. η_{ij} is the inverse of the distance between city i and j. $allowed_k$ is the possible sub-solution that allowed to be used according to taboo list.

After generating new solutions, the pheromone table is updated based on the quality of the newly generated solutions. The pheromone update process follows the calculation shown in Equation (2).

$$\tau_{ij}(t) = (1-\rho)\tau_{ij} + \rho \sum_{k=1}^{m} \Delta\tau_{ij}^{k}. \quad (2)$$

$$\Delta\tau_{ij}^{k} = \frac{1}{L^{k}}. \quad (3)$$

$\Delta\tau$ k in Equation 2 is calculated as Equation 3; where Lk can be one of the follows: length of the tour of kt h ant, length of the best tour in current iteration, or length of the best solution acquired so far. In ACS, Lk is the length of the best tour at the start of the algorithm.

The last step in the main body is local search. Local search is used to improve the quality of the solution that has been acquired. In the case of TSP, the local search step can be employed from the popular local search for TSP, such as 2-opt, 3-opt, and Lin-Kernighan.

2.2. ACO for Clustering

In 2004, Shelokar et al [26] introduced the use of ACO for clustering problem. The idea is to encode the solution for clustering problem into a string representation; each string element represent the sample data and its content represent the cluster number which the sample data assigned to. Each ants in the ACO then build a solution based on the string representation. Table 1 shows the matrix representation of the solution strings for dataset that has $N = 8$ sample data and $K = 3$ cluster numbers, using $M = 5$ ants.

Table 1. Matrix representation example for solution strings in ACO for clustering

S_m	N 1	2	3	4	5	6	7	8
S_1	2	1	2	2	3	1	3	2
S_2	1	3	2	1	1	2	3	3
S_3	1	2	3	3	3	1	2	2
S_4	3	1	1	3	1	2	1	3
S_5	2	3	1	2	3	3	2	1

The general step of the ACO for clustering follows the same step as shown in algorithm 1. The first step in main body is to generate new solutions. In ACO for clustering, an ants can choose a new cluster number in two ways, based on a pre-determined value q0. The first way is to choose the cluster having maximum amount of pheromone among other cluster number in the same data sample. The illustration of the matrix representation of the pheromones is given in Table 2 and Table 3. Second way is to choose the cluster number with probability given in Equation (2), where pi j is the probability of cluster number j to be chosen in sample data i. The first way is analogical to the exploitation process used in ACS [29]. On the other hand, the second way is analogical to biased exploration in ACS. In this step, each ants update the pheromone matrix by using Equation (2).

The second step in the main body is updating the pheromone trail. In this step, only the best ants update the pheromone matrix. The update process is similar to the pheromone update process in the first step except that it use different evaporation rate α shown in Equation (5). The value of Δτi j is also calculated similarly, as shown in Equation (3). Here Fk is the quality of the kt h solution.

$$p_{ij}^{k} = \frac{\tau_{ij}}{\sum_{k=1}^{K} \tau_{ij}}, j = 1...K \tag{4}$$

$$\tau_{ij}(t) = (1-\alpha)\tau_{ij} + \alpha \sum_{k=1}^{m} \Delta\tau_{ij}^{k}. \tag{5}$$

$$\Delta\tau_{ij}^{k} = \frac{1}{F^{k}}. \tag{6}$$

The pheromone is stored in the matrix as shown in Table 2. In order to use the pheromone in Equation (5), the pheromones must be normalized so that for each sample data, the pheromone concentration is summed to 1. Table 3 show the example of normalized pheromone matrix.

Table 2. Example of pheromone matrix in ACO for clustering

Sample	Clusters 1	2	3
1	0.014756	0.015274	0.009900
2	0.015274	0.009900	0.014756
3	0.015274	0.014756	0.009900
4	0.009900	0.015274	0.014756
5	0.014756	0.015274	0.009900
6	0.009900	0.014756	0.015274
7	0.009900	0.020131	0.009900
8	0.015274	0.014756	0.009900

The third step in the main body is to perform local search. Local search is actually a specific technique applicable only for TSP to improve the quality of solutions. To adapt the local search paradigm, ACO for clustering use operator similar to mutation operator in Genetic Algorithm. This local search step is applied only to 20% of the total ants with best solution quality. This local search step starts by generating random number for each sample data in a solution. If there are sample data with random number below a pre-determined threshold value p_{ls}, then its cluster number must be changed to different cluster number.

Table 3. Example of normalized pheromone matrix in ACO for clustering

Sample Data	Clusters		
	1	2	3
1	0.3695	0.3825	0.2479
2	0.3825	0.2479	0.3695
3	0.3825	0.3695	0.2479
4	0.2479	0.3825	0.3695
5	0.3695	0.3825	0.2479
6	0.2479	0.3695	0.3825
7	0.2479	0.5041	0.2479
8	0.3825	0.3695	0.2479

3. PROPOSED ALGORITHM

3.1. The Concept

FACOC improves the ACO computation time by using pattern reduction. Pattern reduction seeks for redun- dant computations that usually occurred in heuristic-based optimization algorithm and cut it to reduce the computation time [30-32]. Pattern reduction reduces the computation time significantly but resulting a little degradation in the solution quality.

In the case of ACO for clustering problem [26], similar redundant computation to what Tseng et al found [32] is observed. As the iteration grows, the pheromone of sample data for certain cluster number become more and more intensive. This fact drives ants to choose the same cluster number in almost certain probability and thus resulting a redundancy in computation. The redundancy even grows more and more massive as the algorithm reach its convergence. Cutting this redundancy will reduce the computation time significantly.

3.2. Algorithm of FACOC

In order to cut the redundancy, the algorithm needs to detect if redundant repetition occurred. It is done by keeping track each time a cluster number is chosen for each sample data. The tracker is recorded in a matrix v with dimension $N \times K$, where N is the number of sample data and K is the total number of clusters. Table 4 shows the example of initial state of matrix v for dataset that has 8 sample data and 3 cluster numbers.

Each time a cluster number is chosen for each sample data, the corresponding value in matrix v is incremented by 1. For example, if an ant choose 1 as cluster number for the second sample data, the value of matrix v in row 2 column 1 is incremented by 1.

Table 4. Example of initial state of matrix v

Sample Data	Clusters		
	1	2	3
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0

To cut redundant repetition, a threshold value Ψ is introduced at the start of algorithm. The value of Ψ is set to $IM\lambda$. I is the number of iteration considered before a cluster number become common. The parameter I determines the iteration point when cluster numbers is starting to become common. M is the

total number of ants. λ is the value that control the percentage value of Ψ to be used. Whenever a value in matrix v exceed Ψ, the corresponding cluster number become common cluster number for the related sample data. The existence of common cluster number recorded to vector ζ , which size is equals to N. The corresponding value of ζ is updated to 1 whenever a cluster number become common and 0 elsewhere. For each sample data that corresponding ζ value is 1, all ants are forced to choose the common cluster number, skipping any probability calculation. This modified cluster number choosing process cuts redundant probability calculation and thus decreasing the computation time. The modified cluster number choosing process is explained by Equation (7); where $C_{common,n}$ is cluster number that already become common on nth sample data; $C_{max\rho,n}$ is cluster number that has the most pheromone in n^{th} sample data. Beside the cluster number choosing process, the local search process is also affected by vector ζ . If the value of ζ for a sample data is 1, it will never be affected by local search process anymore. The complete process of FACOC is explained compactly in algorithm 2.

$$p_{ij}^{k} = \begin{cases} C_{common,n} & if \zeta_n = 1 \\ C_{max\rho,n} & if q = q_0 \\ C_{kn} \ with \ probability \ \frac{\tau_{kn}}{\sum_{k=1}^{K} \tau_{kn}} & otherwise \end{cases} . \quad (7)$$

Algorithm 2 Pseudocode for FACOC algorithm

```
initialize
while iteration < max iteration do
    update matrix v
    check if a cluster number become common
    generate new solutions
    update pheromone
    local search
end while
```

4. PERFORMANCE ANALYSIS

4.1. Research Environment

The research is conducted on a computer with Intel Core i7 processor and 4GB of random access memory (RAM). The operating system used is Windows 7 and the program is built using MATLAB programming language.

The datasets used in this research obtained from UCI machine learning website [33]. Five datasets taken from UCI machine learning website to be used in this research: Iris [34], Wine [35], Parkinsons[36], Connectionist Bench (Sonar, Mines versus Rocks) [37], and Haberman's Survival [38]. The description of the datasets are shown in Table 5.

Table 5. Description of the datasets

Dataset	Number of Pattern	Number of Clusters	Number of Attributes
Iris	150	3	4
Wine	178	3	13
Parkinsons	195	2	22
Connectionist Bench (Sonar, Mines versus Rocks)	208	2	60
Haberman's Survival	306	2	3

4.2. Parameter Initialization

For this experiment, the initialization for parameter used in ACO for clustering and FACOC is shown in Table 6. Note that parameter λ is exclusive to FACOC. This parameter values are chosen because they give optimum result to the algorithm.

As for parameter I and maximum iteration, it determined differently for each datasets. The maximum iter ation is determined according to the number of iteration when the basic ACO usually obtain optimal solution. The parameter I is set around 20% of the maximum iteration. The experiment shows that set I to the number much less than 20% significantly decrease the quality of solution. In contrast, set it much more than 20% makes the decreasing in computation time much less significant. The maximum iteration and value of I is shown in Table 7.

4.3. Performance Evaluation

In this research, sum squared error formula is used to evaluate the quality of solutions. The sum squared error formula is shown in Equation (8), where x_{iv} is the attribute value of v dimension of i^{th} sample data; m_{jv} is the v^{th} dimension value of the centroid for j^{th} cluster; w_{ij} is weight that shows if the i^{th} sample data belongs to j^{th} cluster; the value of w_{ij} is 1 if i^{th} sample data belongs to j^{th} cluster and 0 otherwise.

Table 6. Initialization of parameter for ACO for clustering and FACOC

Parameter	Value
Number of Ants (M)	25
Local evaporation rate (ρ)	0.9
Global evaporation rate (ρ)	0.9
q_0	0.98
p_{ls}	0.01
λ	1

Table 7. Initialization of parameter I and maximum iteration for raw dataset

Dataset	I	Max Iteration
Iris	120	600
Wine	210	800
Parkinsons	80	400
Connectionist Bench (Sonar, Mines versus Rocks)	80	400
Haberman's Survival	100	500

$$F(w,m)=\sum_{j=1}^{K}\sum_{i=1}^{N}\sum_{v=1}^{n} w_{ij}\|\mathbf{x_{iv}}-\mathbf{m_{jv}}\| \quad (8)$$

This research focus on decreasing the computation time taken by ACO for clustering in significant value. Beside that, the quality of the solution produced as shown in Equation (8) also must below the tolerance value of 5%. Thus it resulting a system with significant decreasing of computation time and tolerable quality degradation.

4.4. Initial Experimental Analysis

To confirm that FACOC is actually reduce the computation time, initial experiment on it is conducted to analyze it. In the initial experiment, FACOC and basic ACO for clustering are run and their computation time taken over iteration are plotted together. The plot can be seen in Figure 1. The red line shows the computation time taken for FACOC while the blue line belong to basic ACO for clustering. For this initial experiment, the dataset used is Iris with parameter $I = 120$ and maximum iteration is 600.

It can be seen from Figure 1 that at first, the computation time taken for FACOC grows larger than the basic ACO for clustering. This behavior has been expected as FACOC inject additional command in the program. However, an interesting behavior happen when the cluster number that become common is increasing. This behavior starts from about 120th iteration, which is the value of parameter I. Starting from that point, the computation time of FACOC growing significantly slower than basic ACO for clustering. At the 280th iteration, the FACOC even catch up the basic ACO for clustering and keep growing slower. As the result, starting from that point, the computation time taken for FACOC is faster than basic ACO for clustering.

4.5. Experimental Result

The experimental result is shown in Table 8. From Table 8, it can be seen that across different dataset, the computation time of FACOC is decreased by significant differences of 27.586% in average. On the other hand, the quality is decreased only in small percentage of 3.0352% in average.

Table 8. Experimental result on raw datasets

Dataset	I	Max Iteration
Iris	120	600
Wine	210	800
Parkinsons	80	400
Connectionist Bench (Sonar, Mines versus Rocks)	80	400
Haberman's Survival	100	500

The result shown in Table 8 use the raw data given in each dataset. This approach may be biased because the scale of the value vary between the datasets. To overcome this problem, the performances over normalized dataset is also recorded. The normalized dataset values is calculated as shown in Equation (9); all of them scaled between 1 and 0. In Equation (9), $argmin\{x(i)\}$ and $argmax\{x(i)\}$ consecutively represents the minimum and maximum attribute value for i^{th} dimension. Because the characteristic of the dataset after normalization is different from the raw dataset, the number of maximum iteration is also determined differently. The value of parameter I and the maximum iteration for normalized dataset is shown in Table 9. As for the result, it is recorded as seen in table 10. From Table 10, it could be seen that the scale indeed affect the result. Despite of that, FACOC still managed to output significant differences in computation time by -28.595% in average. The degradation of the quality is still retained in small value also by 1.6569% in average.

Table 8. Experimental result on raw datasets

Dataset	I	Max Iteration
Iris	120	600
Wine	100	500
Parkinsons	80	400
Connectionist Bench (Sonar, Mines versus Rocks)	800	160
Haberman's Survival	200	1000

The result shown in Table 8 use the raw data given in each dataset. This approach may be biased because the scale of the value vary between the datasets. To overcome this problem, the performances over normalized dataset is also recorded. The normalized dataset values is calculated as shown in Equation (9); all of them scaled between 1 and 0. In Equation (9), $argmin\{x(i)\}$ and $argmax\{x(i)\}$ consecutively represents the minimum and maximum attribute value for i^{th} dimension. Because the characteristic of the dataset after normalization is different from the raw dataset, the number of maximum iteration is also determined differently. The value of parameter I and the maximum iteration for normalized dataset is shown in Table 9. As for the result, it is recorded as seen in Table 10. From Ttable 10, it could be seen that the scale indeed affect the result. Despite of that, FACOC still managed to output significant differences in computation time by -28.595% in average. The degradation of the quality is still retained in small value also by 1.6569% zn average.

$$\hat{x}_{iv} = \frac{x_{iv} - argmin\{x(i)\}}{argmax\{x(i)\} - argmin\{x(i)\}} \tag{9}$$

Table 9. Initialization of parameter *I* and maximum iteration for normalized dataset

Dataset	ACO for Clustering		FACOC		Difference	
	Avg. Time	Avg. Quality	Avg. Time	Avg. Quality	Time	Quality
Iris	1.139	7.0129	0.762	7.1866	-33.13%	2.48%
Wine	0.953	49.0618	0.666	50.0867	-30.14%	2.09%
Parkinsons	0.685	98.8003	0.508	100.3685	-25.81%	1.59%
Connectionist Bench (Sonar, Mines versus Rocks)	1.462	446.1638	1.1	446.7613	-24.80%	0.13%
Haberman's Survival	2.156	25.341	1.528	25.8473	-29.10%	2.00%
Average					-28.60%	1.66%

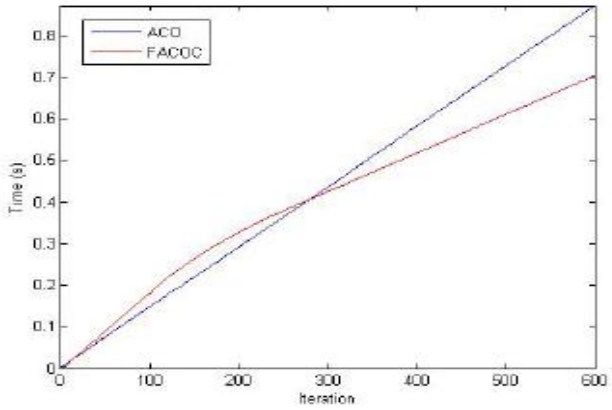

Figure 1. Plot of time vs iteration for FACOC and ACO for clustering algorithm

ACKNOWLEDGEMENT

This work was supported in part by the Ministry of Science and Technology, Taiwan, R.O.C., under grants MOST 106-2218-E-151-003-.

REFERENCES

[1] X. Shi, W. Wang, and C. Zhang, "An empirical comparison of latest data clustering algorithms with state-of-the-art," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 5, no. 2, pp. 410–415, 2017.

[2] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*. Springer, 2006, pp. 25–71.

[3] D. Cai, X. He, and J. Han, "Locally consistent concept factorization for document clustering," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 23, no. 6, pp. 902–913, 2011.

[4] R. Forsati, M. Mahdavi, M. Shamsfard, and M. R. Meybodi, "Efficient stochastic algorithms for document clustering," *Information Sciences*, vol. 220, pp. 269–291, 2013.

[5] M. Verma, M. Srivastava, N. Chack, A. K. Diswar, and N. Gupta, "A comparative study of various clustering algorithms in data mining," *International Journal of Engineering Research and Applications (IJERA)*, vol. 2, no. 3, pp. 1379–1384, 2012.

[6] E. Hancer, C. Ozturk, and D. Karaboga, "Artificial bee colony based image clustering method," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, 2012, pp. 1–5.

[7] K. Jaferzadeh, K. Kiani, and S. Mozaffari, "Acceleration of fractal image compression using fuzzy clustering and discrete-cosine-transform-based metric," *Image Processing, IET*, vol. 6, no. 7, pp. 1024–1030, 2012.

[8] L. Peng, L. He, X. Yang, and K. Wang, "Application of improved fuzzy clustering method in the image segmentation," in *Computational Intelligence and Design (ISCID), 2012 Fifth International Symposium on*, vol. 2. IEEE, 2012, pp. 61–64.

[9] M. Abedini, M. A. M. Said, and F. Ahmad, "Clustering approach on land use land cover classification of landsat tm over ulu kinta catchment," *World Appl Sci J*, vol. 17, no. 7, pp. 809–817, 2012.

[10] A. Ahmad and S. F. Sufahani, "Analysis of landsat 5 tm data of malaysian land covers using isodata clustering technique," in *Applied Electromagnetics (APACE), IEEE Asia-Pacific Conference on*. IEEE, 2012, pp. 92–97.

[11] D. I. Moody, S. P. Brumby, J. C. Rowland, and G. L. Altmann, "Land cover classification in multispectral imagery using clustering of sparse approximations over learned feature dictionaries," *Journal of Applied Remote Sensing*, vol. 8, no. 1, pp. 084 793–084 793, 2014.

[12] J. Senthilnath, S. Omkar, V. Mani, N. Tejovanth, P. Diwakar, B. Shenoy *et al.*, "Hierarchical clustering algorithm for land cover mapping using satellite images," *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, vol. 5, no. 3, pp. 762–768, 2012.
[13] G. F. Elhadi, R. Farouk, and A. T. Issa, "Protein sequence for clustering dna based on artificial neural networks," *International Journal of Computer Science Issues*, vol. 9, no. 1, pp. 161–167, 2012.
[14] P. Stegmaier, A. Kel, E. Wingender, J. Borlak, and I. Ioshikhes, "A discriminative approach for unsupervised clustering of dna sequence motifs," *PLoS Computational Biology*, vol. 9, no. 3, p. e1002958, 2013.
[15] S. Shariff, H. A. Maad, N. N. A. Halim, and Z. Derasit, "Determining hotspots of road accidents using spatial analysis," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 9, no. 1, pp. 146–151, 2018.
[16] E. Sun, C. Wang, and F. Tian, "A survey on clustering routing protocols based on pso in wsn," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 12, no. 7, pp. 5484–5490, 2014.
[17] N. Phing, M. Warip, P. Ehkan, and S. Teo, "Reducing total power consumption and total area techniques for network-on-chip through disable cores and routers based on clustering method," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 10, no. 2, pp. 514–520, 2018.
[18] M.-C. Chiang, C.-W. Tsai, and C.-S. Yang, "A time-efficient pattern reduction algorithm for k-means clustering," *Information Sciences*, vol. 181, no. 4, pp. 716–731, 2011.
[19] A. Kumar, M. Kiran, and B. Prathap, "Verification and validation of mapreduce program model for parallel k-means algorithm on hadoop cluster," in *Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on*. IEEE, 2013, pp. 1–8.
[20] J. C.-W. Lin, L. Yang, P. Fournier-Viger, T.-P. Hong, and M. Voznak, "A binary pso approach to mine high-utility itemsets," *Soft Computing*, vol. 21, no. 17, pp. 5103–5121, Sep 2017.
[21] A. Mexicano, R. Rodrguez, S. Cervantes, P. Montes, M. Jimnez, N. Almanza, and A. Abrego, "The early stop heuristic: A new convergence criterion for k-means," *AIP Conference Proceedings*, vol. 1738, no. 1, p. 310003, 2016.
[22] C.-W. Tsai, K.-W. Huang, C.-S. Yang, and M.-C. Chiang, "A fast particle swarm optimization for clustering," *Soft Computing*, vol. 19, no. 2, pp. 321–338, 2014.
[23] C.-W. Tsai, T.-Y. Lin, M.-C. Chiang, C.-S. Yang, and T.-P. Hong, "Continuous space pattern reduction for genetic clustering algorithm," in *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*. ACM, 2012, pp. 1475–1476.
[24] M. Wang, W. Zhang, W. Ding, D. Dai, H. Zhang, H. Xie, L. Chen, Y. Guo, and J. Xie, "Parallel clustering algorithm for large-scale biological data sets," *PloS one*, vol. 9, no. 4, 2014.
[25] J. Zhang, G. Wu, X. Hu, S. Li, and S. Hao, "A parallel k-means clustering algorithm with mpi," in *Parallel Architectures, Algorithms and Programming (PAAP), 2011 Fourth International Symposium on*. IEEE, 2011, pp. 60–64.
[26] P. Shelokar, V. K. Jayaraman, and B. D. Kulkarni, "An ant colony approach for clustering," *Analytica Chimica Acta*, vol. 509, no. 2, pp. 187–195, 2004.
[27] M. Dorigo and T. Stützle, "Ant colony optimization: overview and recent advances," in *Handbook of metaheuristics*. Springer, 2010, pp. 227–263.
[28] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 26, no. 1, pp. 29–41, 1996.
[29] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 53–66, 1997.
[30] A. S. Girsang, C.-W. Tsai, and C.-S. Yang, "A fast bee colony optimization for traveling salesman problem," in *Innovations in Bio-Inspired Computing and Applications (IBICA), 2012 Third International Conference on*. IEEE, 2012, pp. 7–12.
[31] S.-P. Tseng, C.-W. Tsai, M.-C. Chiang, and C.-S. Yang, "Fast genetic algorithm based on pattern reduction," in *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*. IEEE, 2008, pp. 214–219.
[32] S.-P. Tseng, C.-W. Tsai, M.-C. Chiang, and C.-S. Yang, "A fast ant colony optimization for traveling salesman problem," in *Evolutionary Computation (CEC) 2010 IEEE Congress on*. IEEE, 2010, pp. 1–6.
[33] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml
[34] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.
[35] M. Forina *et al.*, "Arvus-an extendible package for data exploration, classification and correlation," *Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, Italy*, 1991.
[36] M. A. Little, P. E. McSharry, E. J. Hunter, J. Spielman, and L. O. Ramig, "Suitability of dysphonia measurements for telemonitoring of parkinson's disease," *Biomedical Engineering, IEEE Transactions on*, vol. 56, no. 4, pp. 1015–1022, 2009.
[37] R. P. Gorman and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural networks*, vol. 1, no. 1, pp. 75–89, 1988.
[38] S. J. Haberman, "Generalized residuals for log-linear models," in *Proceedings of the 9th International Biometrics Conference*, 1976, pp. 104–122.